# TechPartHub: 电脑配件一站式购物平台

## 小组成员

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **专业** | **工作内容概述** |
| **1** | **202200800092** | **郭富政** | **计算机澳** | **负责项目的整体规划、分工协调和进度跟踪。同时也负责测试工作，确保软件的质量和性能。** |

## 项目概述

### 2.1项目背景

随着信息技术的快速发展，电脑已成为日常生活和工作中不可或缺的工具。用户对电脑性能的需求日益增长，导致市场上对高质量电脑配件的需求持续增加。

为了满足这一市场需求，我们决定开发一个基于最新Web技术的电脑配件销售平台。该平台采用Vue.js作为前端开发框架，以其高效的数据驱动视图和组件化的开发方式，提供丰富且用户友好的界面。后端选择SpringBoot，凭借其轻量级、快速开发的特性，构建稳定且高效的后端逻辑。

### 2.2 项目功能介绍

我们的电脑配件销售平台是一个综合性的电子商务解决方案，使用Vue.js结合Element UI和ECharts进行前端开发，后端采用SpringBoot，数据库使用MySQL。该平台集成了多项先进功能和服务，以满足现代电商需求：

* **用户管理：**平台提供全面的用户管理功能，包括用户注册、登录、个人信息管理等，确保用户体验的个性化和便捷性。
* **单商户商品管理：**商户可以快捷管理自己的商品库存，包括商品上架、编辑、下架等操作。这提高了商品管理的效率和灵活性。
* **支付系统：**集成安全可靠的支付系统，支持多种支付方式，确保交易的安全性和便捷性。
* **发货系统：**提供高效的发货管理系统，包括订单处理、物流跟踪等，确保订单的及时处理和透明追踪。
* **邮件提示系统：**自动发送邮件通知用户关于订单状态的更新，增强用户体验和沟通效率。
* **分页功能：**为了提高用户界面的响应性和数据加载效率，平台采用分页功能来显示商品列表和搜索结果。
* **多角色管理：**支持多角色用户管理，包括普通用户、商户和管理员等，每个角色有其特定的权限和功能，确保平台运行的顺畅和安全。
* **后台管理：**提供强大的后台管理功能，管理员可以进行用户管理、订单管理、商品审核等操作，保证平台的高效运行。
* **统计图表：**利用ECharts展示销售数据和用户行为的统计图表，帮助商户和管理员更好地理解市场趋势和用户需求。
* **日志记录：**实现详细的日志记录功能，用于监控系统操作和维护安全性。
* **图像文件上传：**用户和商户可以上传商品图片和个人头像，增强信息的直观性和互动性。

## 三、项目实现过程和截图

### 3.1**项目使用的技术介绍**

我们的电脑配件销售平台是基于当代领先的Web开发技术构建的。以下是项目中使用的关键技术及其应用：

* **Vue.js：**Vue.js是一个高效的JavaScript框架，用于构建用户界面。它以数据驱动和组件化的方法来简化前端开发，使我们能够创建一个快速且响应灵敏的用户界面。
* **Element UI：**Element UI是一个基于Vue的UI组件库，提供了一系列预制的高质量组件，如按钮、表单、表格等，这些都是构建现代Web应用程序的基础。通过使用Element UI，我们保持了界面的一致性和专业性。
* **ECharts：**ECharts是一个基于JavaScript的开源可视化库，用于生成各种交互式图表。在我们的平台中，ECharts被用于生成销售数据和用户行为分析的图表，为商户和管理员提供了直观的数据洞察。
* **SpringBoot：**SpringBoot是一个流行的Java框架，用于构建高效的企业级应用程序。它简化了Spring应用程序的配置和部署。在我们的平台中，SpringBoot用于后端开发，提供RESTful API，处理数据逻辑，确保应用的性能和安全性。
* **MySQL：**MySQL是一个强大的开源关系型数据库管理系统，用于存储和管理应用数据。我们选择MySQL作为后端数据库，是因为它的稳定性、可扩展性和广泛的社区支持，确保了我们数据处理的高效和可靠。
* **MyBatis-Plus：**MyBatis-Plus 是一个为了简化开发工作而创建的 MyBatis 的扩展库。它内置了多种便捷功能，如自动CRUD操作，通过继承BaseMapper就能够省去许多传统的配置和代码。它还包括强大的条件构造器和分页插件，简化了SQL构建和数据分页的处理。

通过对以上技术的应用，我们的平台的使用体验、开发效率都有不小的提升。同时，我们也在开发的过程中获得了宝贵的经验。

### 3.2**项目的实现过程**

### 数据库设计

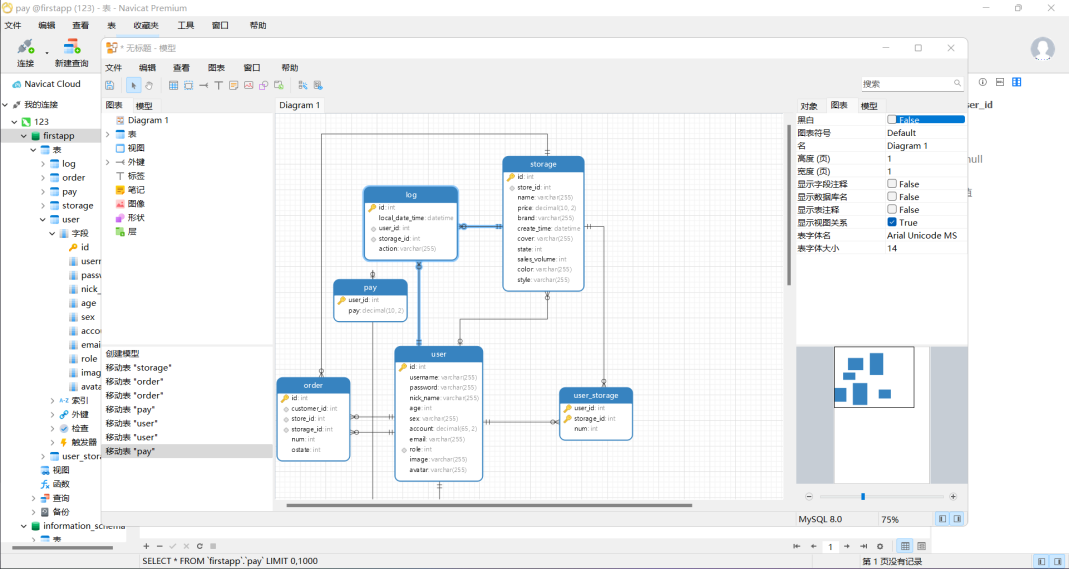


图 1：数据库模型设计图（使用Navicat）

我们的电脑配件销售平台包含了六个主要的数据表：log（日志），order（订单），storage（商品），user（用户），以及user\_storage（商户与商品的关联表）。

* **log（日志）表**：此表用于记录应用中的重要事件，如用户操作、系统错误、事务日志等。字段包括：日志ID、事件类型、时间戳、用户ID（如果适用）、描述等。
* **order（订单）表**：存储用户提交的订单信息。字段包括：订单ID、用户ID、订单状态、总金额等。
* **storage（商品）表**：存储平台上销售的商品信息。字段包括：商品ID、商品名称、价格、描述、图片链接等。
* **user（用户）表：**记录平台用户的信息。字段包括：用户ID、用户名、密码、邮箱、电话、用户类型（如普通用户、商户）等。
* **user\_storage（商户与商品的关联）表：**用于表示商户和他们销售的商品之间的关系。字段包括：关联ID、用户ID（商户）、商品ID等。

数据库的备份文件我们会在作业提交中给出。

### 后端**RESTful API设计**

###### 

图 2：项目结构

首先，我们一起梳理了我们的电脑配件销售平台应该满足哪些需求，进而确定了需要实现的功能，如用户注册登录、商品浏览、订单管理、支付处理等。每个功能都与一个特定的业务逻辑关联，这将决定我们需要构建的API接口。

基于需求分析，我们开始设计RESTful风格的API接口。这意味着我们需要为每种资源（如用户、订单、商品等）定义标准的HTTP动作（GET、POST、PUT、DELETE）。

例如，对于用户管理，API设计如下：

* **GET /user：**获取用户列表
* **POST /user：**创建一个新用户
* **GET /user/{id}：**获取一个特定用户的详细信息
* **PUT /user/{id}：**更新一个特定用户的信息
* **DELETE /user/{id}**：删除一个特定用户

在设计了API接口后，我们开始在Spring框架中使用@Controller注解来实现这些接口。我们为每个业务模型创建了一个控制器类，并在类中定义了处理相应HTTP请求的方法。



图 3：UserController的具体实现 图 4：实现发送邮件提醒功能

得益于mybatis-plus，我们可以直接在controller中使用BaseMapper中提供的一系列原生的CRUD（创建、读取、更新、删除）功能。完善entity和mapper后，我们在application.properties中将后端和MySQL连接起来。值得注意的是，我们还使用了Spring的邮件发送接口。Spring提供了非常好用的JavaMailSender接口实现邮件发送。由于SpringBoot的Starter模块也为此提供了自动化配置，所以在引入了spring-boot-starter-mail依赖之后，会根据配置文件中的内容去创建JavaMailSender实例，因此我们可以直接在需要使用的地方直接@Autowired来引入邮件发送对象。

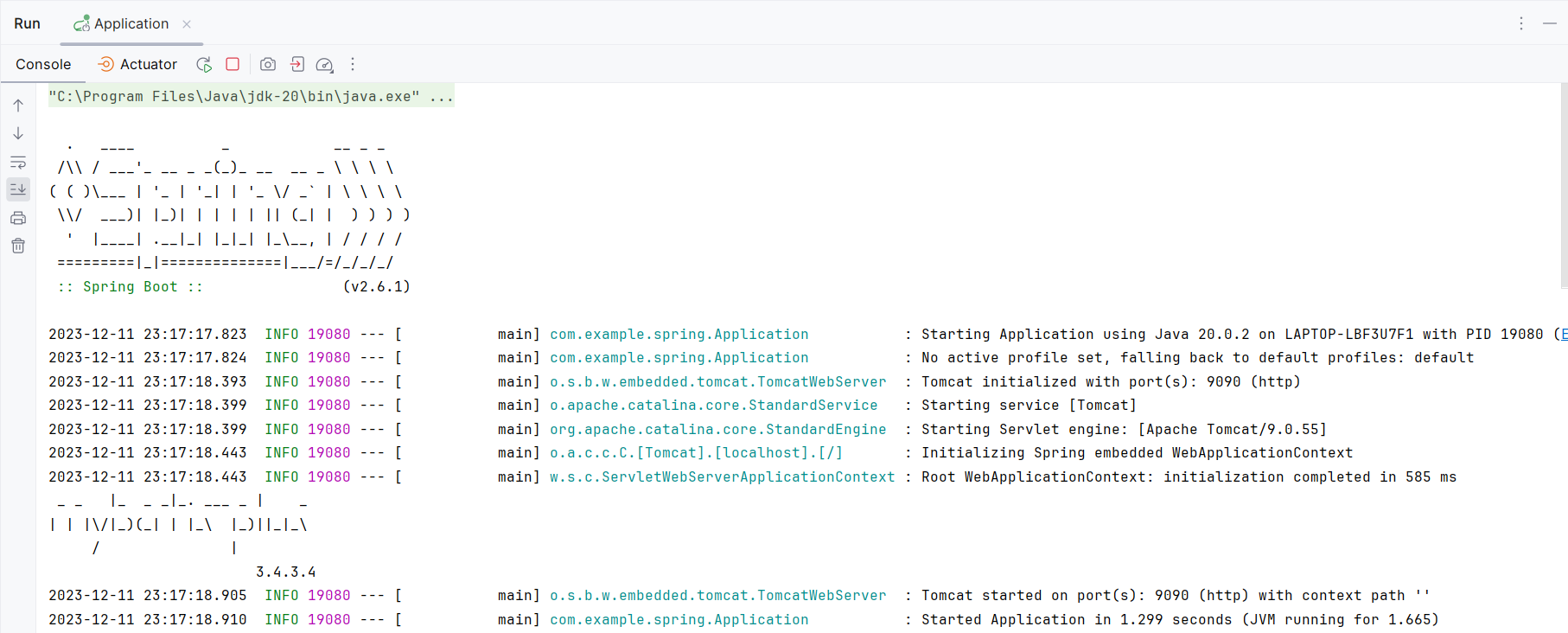


图 5：后端运行截图

应用程序的启动被标记为成功，表明所有的上下文和依赖项都已正确解析，服务器正在监听预设的端口上等待接收请求。这是项目从开发到实际运行的关键一步，验证了后端架构的有效性和应用程序的稳定性。后续，我们进行了详细的功能测试，以确保每个API端点都能够按预期那样响应前端的请求。

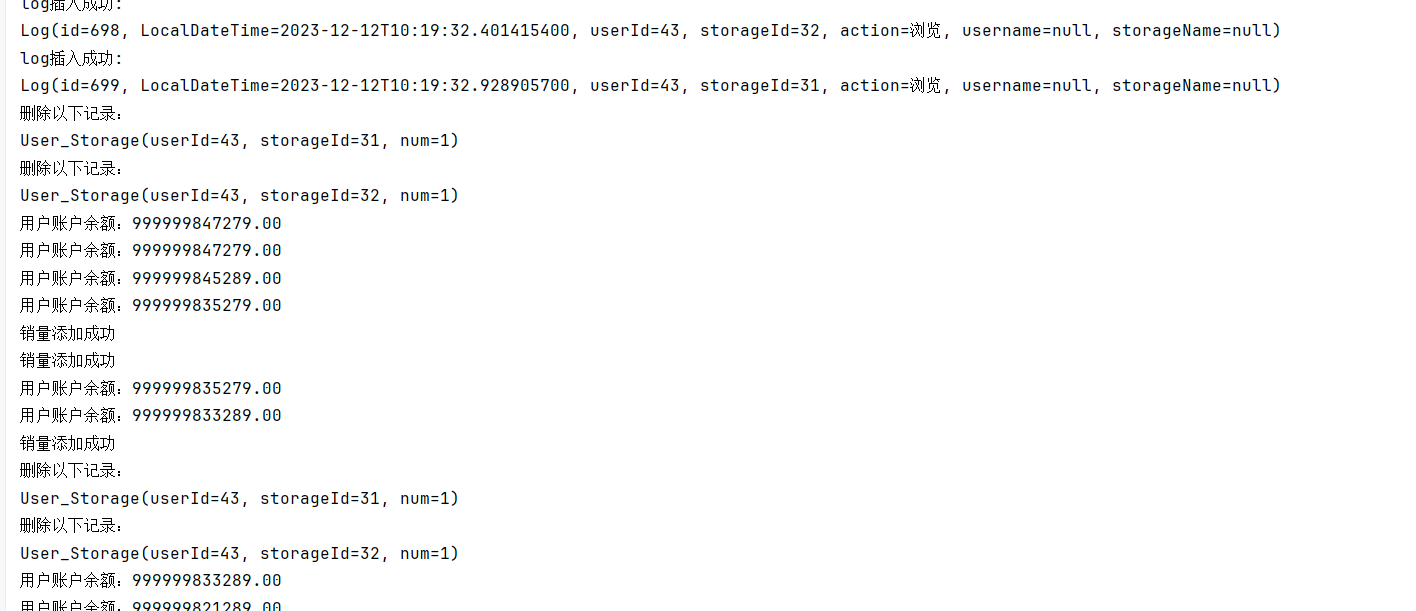


图 6:运行输出（在Controller嵌入输出语句以判断是否按预期运行）

### （三）前端设计

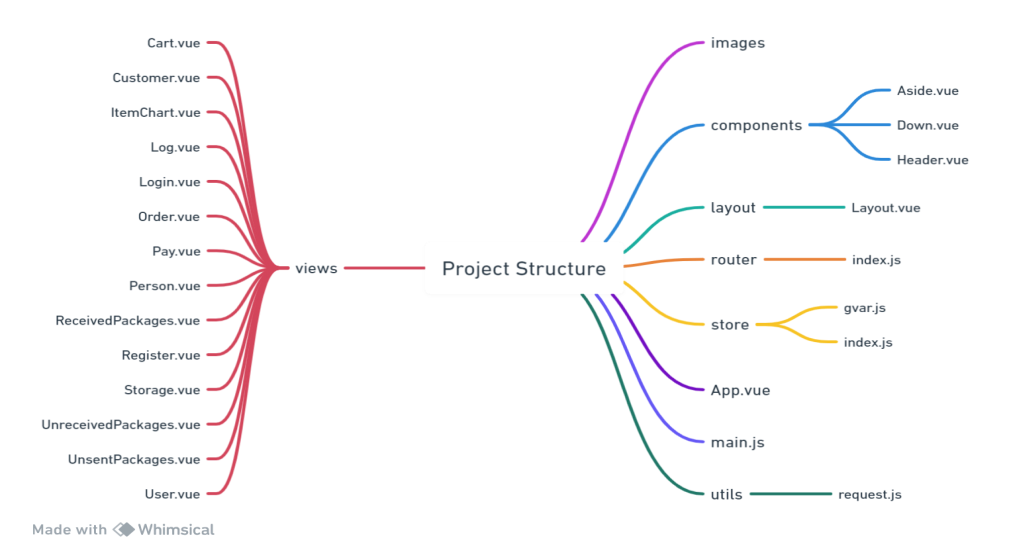


图 7：结构预览

**①数据连接层：**

在我们的Vue项目中，数据连接层是负责与后端服务交互的部分，处理数据的获取、发送和更新。这通常包括调用API接口、处理响应和错误，以及管理前端状态以反映这些数据变化。在我们的项目结构中，数据连接层涉及以下两个部分：

* **utils/request.js:** 这个文件包含用于与后端服务通信的通用函数和设置。它包括设置如axios或其他HTTP客户端的实例，配置请求头，处理请求和响应拦截器等。
* **store 目录(gvar.js 和 index.js)**:这里是Vuex存储，用于管理应用的全局状态。数据连接层在这里体现为与后端API通信来获取或更新数据，然后将这些数据存储在Vuex中，以便在整个应用中共享和访问。

通过对数据连接层的包装，我们可以专注于前端编程，而无需关心后端的具体实现。

**②路由配置层：**

路由层对于构建一个具有多个视图和组件的前端应用程序非常重要。对于这个项目，我们需要为router目录下的index.js文件配置路由，确保应用程序的不同页面（视图）能够通过URL进行访问。路由配置还设置了父子路由关系，这种结构主要用于构建具有嵌套视图的应用程序，如具有统一布局但不同内容的页面。在我们的程序中Layout路由充当一个父路由，以设置网页的主页面。它用于定义一个共同的布局或结构，该结构在多个子页面间共享。通常，这涉及到一个共同的头部、侧边栏、底部等UI组件。在代码中，Layout路由如下定义：



图 8：Layout

子路由定义在children数组中。每个子路由都是一个对象，它指定了一个路径、一个名称以及当该路径匹配时应该渲染的组件。例如user子路由，当用户访问/user路径时，将渲染User组件。因为这个路由是在Layout路由的children数组中定义的，所以User组件将在Layout组件内部渲染。这意味着Layout组件的布局（如头部、侧边栏等）将保持不变，而主内容区域将显示User组件的内容。

同时我们还可以使用Vue Router的导航守卫来控制访问权限，比如在用户未登录时重定向到登录页面；客户无法进入后台管理界面，而管理员能够进入后台管理界面等。

**③公共组件层：**

在我们的 Vue 项目中，公共组件层是一个关键的部分，它包含了多个重用组件，这些组件在整个应用中多次使用，旨在提高我们的开发效率，确保应用的一致性和可维护性。

组件表列如下：

* **Aside.vue -** 侧边导航组件，用于在应用的侧边栏中提供快速的导航功能。
* **Down.vue -** 一个页面组件，用于在应用页脚等位置显示信息。
* **Header.vue -** 顶部导航栏组件，显示应用的标题和顶部菜单项。

公共组件层在我们的 Vue 项目中起着至关重要的作用。通过有效利用这些组件，我们不仅提高了开发效率，还简化了维护的工作量。随着项目的进展，我们扩展和完善这些组件，以满足不断变化的需求。

**④布局层：**

Layout.vue 是本项目的核心布局组件。它负责构建和维护应用程序的整体布局框架，包括导航栏、侧边栏、页脚以及主内容区域的相对位置。此布局设计旨在提供一致且直观的用户体验，确保应用的各个部分协调一致地工作。

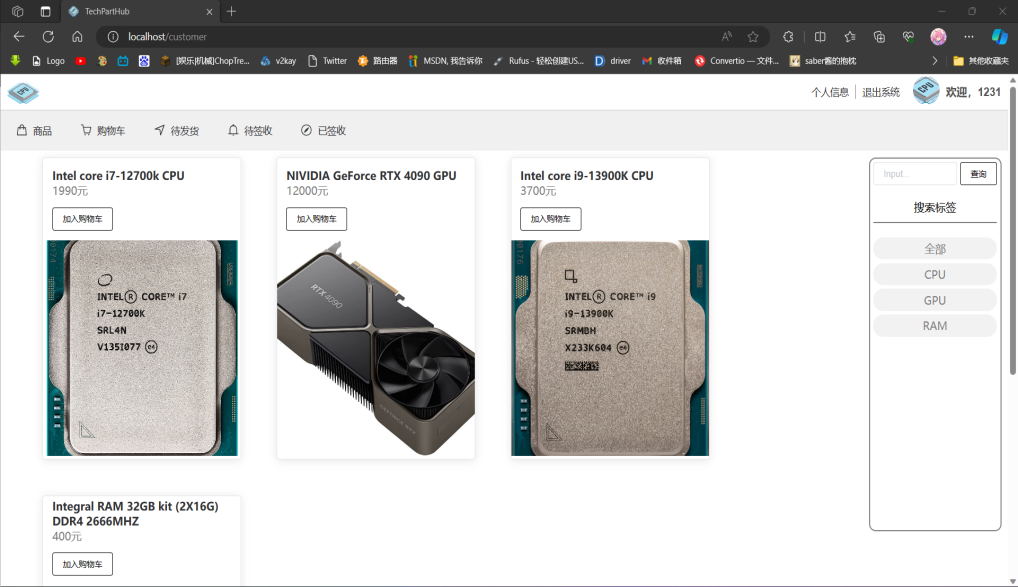


图 9：Layout设计

**⑤视图层：**

我们在设计网页主视图的时候使用了Element UI组件库，包括其中的card、table、button等组件。在统一风格的同时也简化了我们的开发难度，同时使设计出来的UI更易于接受。

我们总共创建了14个网页视图，以下是各个视图的简介：

| **视图** | **功能描述** | **视图** | **功能描述** |
| --- | --- | --- | --- |
| 登录 (Login.vue) | 用户通过输入用户名和密码来访问应用程序。提供简洁登录表单，包括错误处理和数据验证。 | 注册 (Register.vue) | 允许新用户创建账户。包括基本信息输入表单，提供数据有效性检查。 |
| 统计图表 (ItemChart.vue) | 展示商品的详细信息和统计数据，利用图表和数据可视化工具。 | 购物车 (Cart.vue) | 用户查看选购商品，修改数量或移除商品，提供结账选项。 |
| 订单 (Order.vue) | 显示订单历史和当前状态，包括商品详情、总额和配送状态。 | 支付 (Pay.vue) | 提供安全支付界面，多种支付方式选择。 |
| 用户管理 (User.vue) | 后台部分，仅管理员访问，管理平台注册用户。 | 个人信息 (Person.vue) | 显示用户个人信息，可上传头像。 |
| 商品管理 (Storage.vue) | 商品增删查改操作。 | 日志 (Log.vue) | 查看操作日志。 |
| 已签收订单 (ReceivedPackages.vue) | 查看已签收订单。 | 未发货订单 (UnsentPackages.vue) | 查看未发货订单。 |
| 未签收订单 (UnreceivedPackages.vue) | 查看已发货未签收订单。 | 商品主页面 (Customer.vue) | 浏览所有商品。 |

### （四）网页运行截图

在命令行运行npm run serve后，我们可以看见网页在本地服务器上运行了起来。

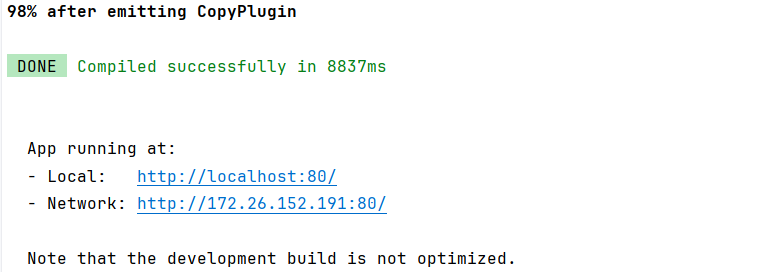


图 10：命令行截图

我们可以在浏览器输入localhost打开我们的网页，同时经过路由配置重定向到localhost/login。

以下是我们的网页截图：

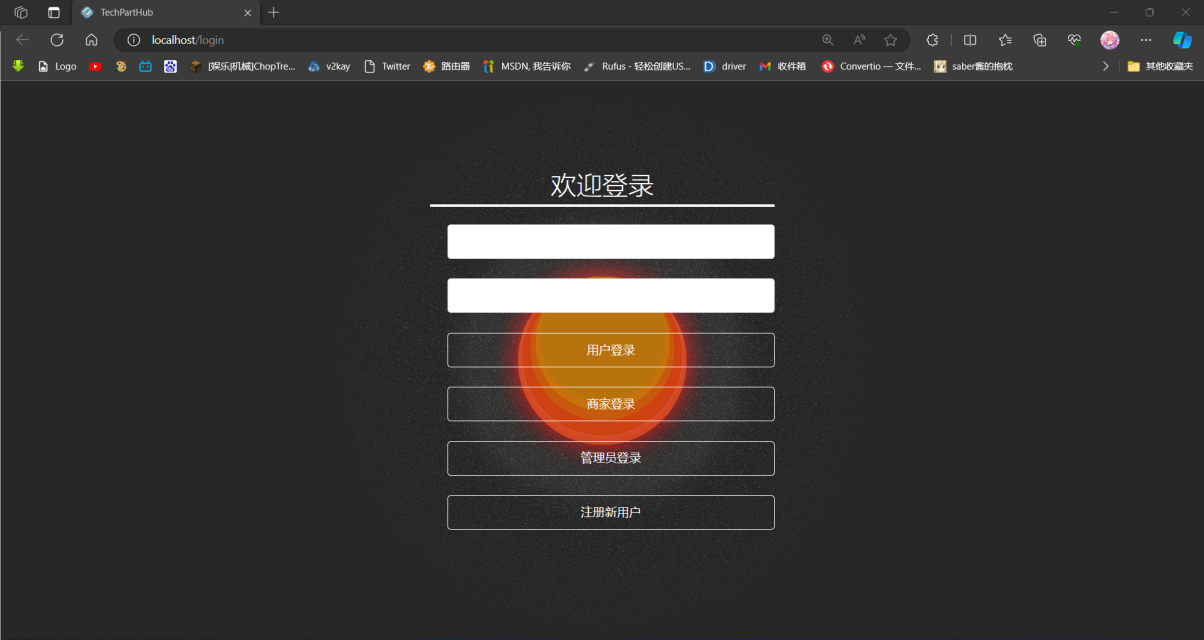


图 11：欢迎界面

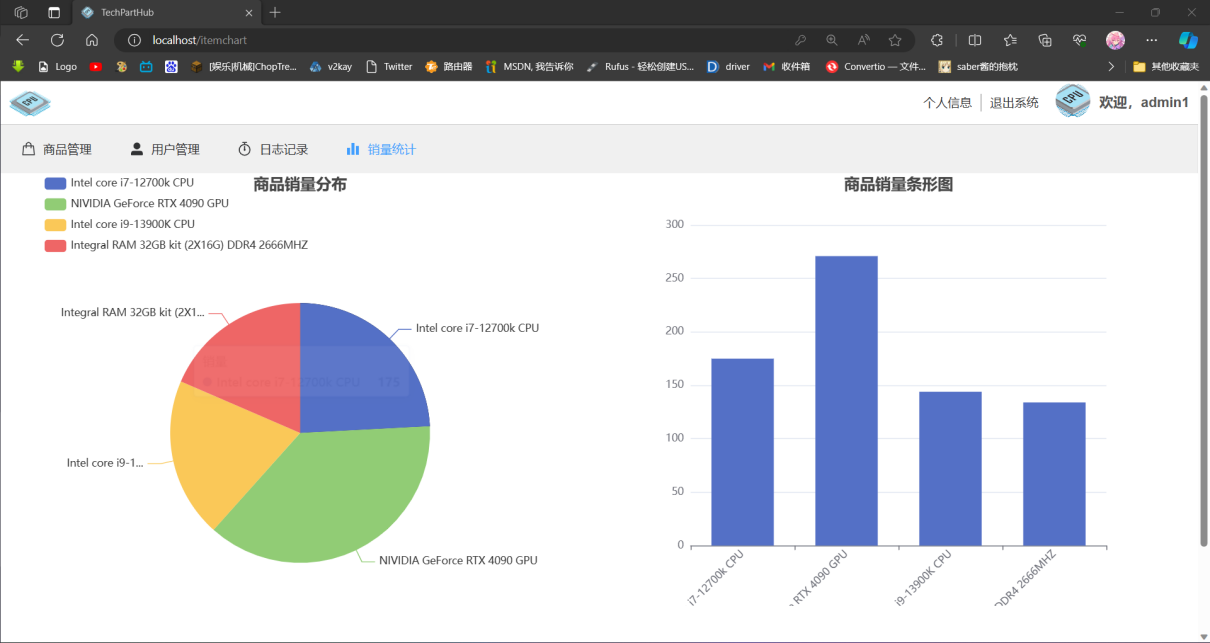


图 12：EChart组件

## 四、进一步优化思考

由于开发时间的限制，我们并没有非常多的时间来打磨我们的作品。因此我们仅仅只是实现了一个购物平台的基本功能，很多功能还没有落地，例如根据用户的浏览记录判断用户的喜好从而进行个性化推荐，客服与客户的实时聊天窗口、打折促销等。

但是我们相信，随着学习的深入，我们掌握的技术栈会大大丰富。届时，开发一个我们心目中的在线购物平台将不在话下。